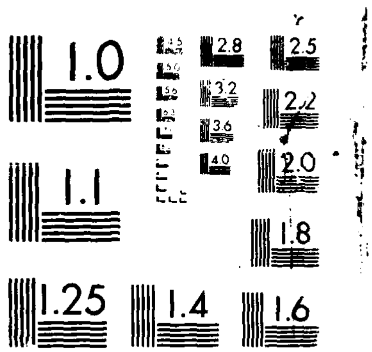
AD-A195 184 SIMMIN - A PROGRAM FOR MINIMIZING FUNCTIONS BY SIMPLEX 1/1
METHODS FOR IBM PC..(U) ARMY MISSILE COMMAND REDSTONE
ARSENAL AL DIRECTED ENERGY DIRE.. J D SMITH ET AL.
UNCLASSIFIED 09 JUL 87 AMSMI/TR-RD-DE-87-4 F/G 12/4 NL


1.0
2.8
2.5
3.2
2.2
3.6
4.0
2.0
1.1
1.8
1.25
1.4
1.6

(2)

TECHNICAL REPORT RD-DE-87-4

DTIC FILE COPY

SIMMIN - A PROGRAM FOR MINIMIZING FUNCTIONS BY SIMPLEX METHODS FOR IBM PC

Jerry D. Smith
Miles E. Holloman
William F. Otto
Directed Energy Directorate
Research, Development,
and Engineering Center

AD-A195 184

JULY 1987

U.S. ARMY MISSILE COMMAND
Redstone Arsenal, Alabama 35898-5000

Approved for public release; distribution is unlimited.

SMI FORM 1021, 1 AUG 85 PREVIOUS EDITION IS OBSOLETE

88 5 02 008

DISPOSITION INSTRUCTIONS

DESTROY THIS REPORT WHEN IT IS NO LONGER NEEDED. DO NOT RETURN IT TO THE ORIGINATOR.

DISCLAIMER

THE FINDINGS IN THIS REPORT ARE NOT TO BE CONSTRUED AS AN OFFICIAL DEPARTMENT OF THE ARMY POSITION UNLESS SO DESIGNATED BY OTHER AUTHORIZED DOCUMENTS.

TRADE NAMES

USE OF TRADE NAMES OR MANUFACTURERS IN THIS REPORT DOES NOT CONSTITUTE AN OFFICIAL INDORSEMENT OR APPROVAL OF THE USE OF SUCH COMMERCIAL HARDWARE OR SOFTWARE.

ADA195184

SECURITY CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE

Form Approved OMB No. 0704-0188 Exp. Date: Jun 30, 1986

1a. REPORT SECURITY CLASSIFICATION: UNCLASSIFIED

1b. RESTRICTIVE MARKINGS:

2a. SECURITY CLASSIFICATION AUTHORITY:

2b. DECLASSIFICATION/DOWNGRADING SCHEDULE:

3. DISTRIBUTION/AVAILABILITY OF REPORT: Approved for public release; distribution is unlimited

4. PERFORMING ORGANIZATION REPORT NUMBER(S): TR-RD-DE-87-4

5. MONITORING ORGANIZATION REPORT NUMBER(S):

6a. NAME OF PERFORMING ORGANIZATION: Directed Energy Res, Dev, and Eng Ctr

6b. OFFICE SYMBOL (If applicable): AMSMI-RD-DE

7a. NAME OF MONITORING ORGANIZATION:

6c. ADDRESS (City, State, and ZIP Code):
Commander
U.S. Army Missile Command, ATTN: AMSMI-RD-DE
Redstone Arsenal, AL 35898

7b. ADDRESS (City, State, and ZIP Code):

8a. NAME OF FUNDING/SPONSORING ORGANIZATION:

8b. OFFICE SYMBOL (If applicable):

9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER:

8c. ADDRESS (City, State, and ZIP Code):

10. SOURCE OF FUNDING NUMBERS

PROGRAM ELEMENT NO.	PROJECT NO.	TASK NO.	WORK UNIT ACCESSION NO.

11. TITLE (Include Security Classification): SIMMIN - A Program for Minimizing Functions by Simplex Methods for IBM PC

12. PERSONAL AUTHOR(S): Jerry D. Smith, Miles E. Holloman, William F. Otto

13a. TYPE OF REPORT: Final

13b. TIME COVERED: FROM ______ TO ______

14. DATE OF REPORT (Year, Month, Day): 1987/07/09

15. PAGE COUNT: 27

16. SUPPLEMENTARY NOTATION:

17. COSATI CODES

FIELD	GROUP	SUB-GROUP

18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number): Minicomputer, Simplex, Nonlinear programming, Optimal solutions, Optimization methods

19. ABSTRACT (Continue on reverse if necessary and identify by block number)

SIMMIN is a minicomputer program capable of finding a minimum for any function which can be described using Fortran. A simplex method determines values of coefficients which result in a function local minimum. This document briefly describes the methodology and algorithm originally written by Caceci and Cacheris and modified by Holloman and Otto. It also provides a user's guide for use of the Fortran code (which is included), complete with example cases. Conversion to larger computers is straightforward.

20. DISTRIBUTION/AVAILABILITY OF ABSTRACT: ☒ UNCLASSIFIED/UNLIMITED ☒ SAME AS RPT. ☐ DTIC USERS

21. ABSTRACT SECURITY CLASSIFICATION: UNCLASSIFIED

22a. NAME OF RESPONSIBLE INDIVIDUAL: Jerry D. Smith

22b. TELEPHONE (Include Area Code): (205) 876-4329

22c. OFFICE SYMBOL: AMSMI-RD-DE-SD

DD FORM 1473, 84 MAR — 83 APR edition may be used until exhausted. All other editions are obsolete.

SECURITY CLASSIFICATION OF THIS PAGE

SUMMARY

A computer program, SIMMIN, which is capable of finding a minimum (if one exists) for any function which can be described using Fortran, has been developed and tested on an IBM PC with an 8087 math coprocessor. The program is a modification of a Pascal algorithm written by Caceci and Cacheris and modified by Holloman and Otto. A simplex method determines values of initially unknown coefficients which result in a local function minimum.

The user can vary initial estimates of the coefficients, step sizes, maximum number of iterations, and the convergence criteria, of which two are availlable. Final values of the coefficients, an estimate of the function minimum, and the number of iterations required are displayed on screen, while a diagnostic iteration history is recorded on a separate file which can be examined after each execution.

The program can easily be converted for use on larger computers, perhaps allowing the use of single precision. Although divergence is impossible, a price is often paid in more iterations and computing time than with some other traditional minimization methods.

TABLE OF CONTENTS

I. INTRODUCTION

Techniques for minimizing functions include stepwise descent, steepest descent, Newton-Raphson algorithms (and variations), and Marquardt algorithms. However, there are many functions for which these well-known methods are difficult to implement, are computer intensive, and are prone to diverge. Such functions may be nonlinear, partially empirically defined, not continuous in regions of interest, or quite complicated. Inaccurate calculation of derivatives and the lack of good initial estimates for the unknown coefficients are often responsible for the failure of these methods.

The program SIMMIN, the subject of this report, can find a minimum (if one exists) for any function that can be described by the user in Fortran. SIMMIN does this by using a simplex technique to calculate values of initially unknown coefficients which result in a functional minimum. The user should have some familiarity with Fortran so that the subroutine FUNC in the source code shown in the Appendix can be modified for the function of interest. A very simple function has been included for instructional purposes.

II. THE MODEL

The simplex methodology for finding functional minima is used in SIMMIN. The original algorithm was written by Nelder and Mead in 1965 [1] and was implemented in Extended Mercury Autocode on an Orion computer. The starting point for the present study is a Pascal algorithm by Caceci and Cacheris to fit curves to data [2]. Holloman and Otto modified the model by converting to Fortran, introducing a different convergence criterion, to be discussed later, and hard-coding user inputs, such as initial estimates and step sizes, in order to make the model more transparent to the user. They next incorporated their modified model into user-friendly, curve-fitting programs for IBM PC [3,4]. In addition to familiar Fourier, Lagrange and spline methods, data could be fit to any equation selected by the user. The coefficients in the equation were determined by allowing the simplex technique to minimize the nonlinear function sum of squares of residuals as follows:

$$SSR = \sum_{i=1}^{N} [\ Ycalc(X_i, A_1, A_2, \ldots, A_{mm}) - Yexper(X_i\ )\ ]^2 \qquad (1)$$

where A_1, $A_2, \ldots, A_{mm}$ are the unknown coefficients and N is the number of experimental data points, X_i. Least squares fitting to any equation was done without using derivatives or matrix operations. However, it was noted that for most fitting functions, execution times increased greatly with the number of coefficients, above approximately three.

In the present study, the fitting aspects of the model were removed. User control over most inputs was restored since finding a minimum (or minima) usually requires experimentation with initial estimates, step sizes, the maximum number of iterations, and the convergence criteria, of which two are available.

The user is referred to References 1 and 2 for details of the model, but a short introduction is offered here. A simplex is a geometric figure which has

one more vertex than the space in which it is defined has dimensions. For example, the user will find in the Appendix (the source code for program SIMMIN) a function of the following form defined in line numbers 281-284:

$$F = (A-1.)^2 + (B-2.)^2 + (C-3.)^2 \tag{2}$$

The simplex for this function would have four vertices in three-dimensional space. Each vertex is characterized by four values: A,B,C,F. Subroutine SIMPLEX defines the simplex as a two-dimensional array SIMP(I,J), with I being the vertex index and J the dimension index. It should be clear that for our example, values A = 1, B = 2, C = 3 yield the one and only function minimum F = 0.

The principal of the simplex method is to move and reshape the simplex in ways resulting in decreased values of the response function as iterations proceed. Each iteration involves combinations of reflection, expansion, contraction, and shrinkage of the vertices according to logical rules, and then testing the newly calculated function value against the previous value. If the new value is better (meaning lower) than the previous one, then the vertices are accepted and the simplex proceeds in the same general direction. If not, then the vertex changing logic proceeds in an orderly, efficient search or a new direction.

III. COMPUTER IMPLEMENTATION

The present study was done on an IBM PC equipped with an 8087 coprocessor using Ryan-McFarland Fortran (V2.11). This Fortran creates very efficient executable code at the expense of longer compilation times. Double precision was used for all reals by appropriately placed implicit double precision statements. The user can easily remove the double precision if SIMMIN is used on fast computers with large word size, where large numbers of iterations and truncation/round-off errors are not likely to cause problems using single precision.

The primary modification the user will make is to substitute a function of interest for the demonstration function in subroutine FUNC (see the Appendix, Lines 272-287). The program requires that the coefficients enter the subroutine as an array COEF(I), I = 1,...,MM, where MM is the number of coefficients to be determined. The user is then free to convert to a more convenient notation. Perhaps the function to be minimized is available as a function subprogram or a subroutine which would then be called from within subroutine FUNC. The calculated value of the function must be returned from FUNC as COEF(MM+1).

Constraints can be added in the form of Fortran statements directing the program flow to statement 999 in FUNC, often in the form of IF statements restricting the coefficients or function to certain regions. Note what happens then: the function value COEF(MM+1) is returned as the previous value plus a very large number 1.D30. Most calls of subroutine FUNC from within subroutine SIMPLEX are followed by IF tests to determine if the new trial function value RNEXT(MM+1) is less than the previous value, referred to in subroutine simplex as SIMP(MM+1,IH(MM+1)). It certainly will not be if a constraint condition has been exceeded and the logic will guide the simplex away from this region.

An important additional word about constraints: the user may well have them without being aware of them for awhile. The simplex moves about in hyperspace without regard to such problems as attempting to take logarithms of zero or of negative numbers, dividing by zero, or attempting to evaluate functions (such as factorials, gamma functions, etc.) in regions leading to computer underflow or overflow. The user should protect against all such possibilities by adding appropriate statements to direct the program to statement 999, or to do the equivalent action described after statement 999 somewhere else.

A. Input Parameters

Values supplied by the user during interactive operation of SIMMIN are shown within boxes in Figures 1, 2, and 3.

1. MM - The number of coefficients to be calculated, 3 for the example in Equation (2).

2. MAXITR - The maximum number of iterations allowed. This may be only approximate since the program now counts every trial set of vertices as an iteration, regardless of whether the function value increases or decreases. Therefore, the number of iterations shown as part of the results could slightly exceed MAXITR.

3. Initial Estimates of the Coefficients - The user must insure that their order conforms to the definition of coefficients in subroutine FUNC. One should always experiment with these initial estimates to see the effect on results.

4. Convergence Criterion Option

ICON = 1 -- The program compares relative errors of calculated coefficients with a limit entered by the user, typically between 1.D-05 and 1.D-04. If every relative error is less than the limit, then convergence is obtained. Before defining this error, recall for Equation (2) that there are four vertices for the simplex: A, B, C, F, and that each vertex is characterized by four values: A, B, C, F. The simplex is defined as a two-dimensional array SIMP(I,J), with I being the vertex index and J being the dimension index. The error is defined as follows (Appendix, Line 231):

```
ERROR(I)=(SIMP(I,IH(I))-SIMP(I,L(I)))/SIMP(I,IH(I))          (3)
```

subroutine ORDER has provided IH(I) and L(I), the indices of the highest and lowest values, respectively, currently associated with vertex I. The value of ERROR(MM+1), the function error, does not seem to decrease to nearly as small as the coefficient errors ERROR(I), I=1,MM even though inspection of latter iterations in file ITER.OUT shows a minimum is close to being found, by either convergence criterion. Therefore, the error comparisons were restricted to the coefficients only.

ICON = 2 -- This convergence option allows the program to keep a count of the number of consecutive times that new trial vertices have been calculated without reducing the value of the function. If this count exceeds (MM+1)*3 then convergence is obtained. It was arrived at through experience

with nonlinear functions in fitting applications, and relieves the user from choosing acceptable limits for errors [3,4]. The simplex has usually found a minimum. However, it may be trapped near some unusual topological feature, perhaps moving up and down a valley where coefficient values change but the function value does not. When dealing with new, unfamiliar, complicated functions, it is always advisable to experiment with initial estimates and step sizes and to examine at least the latter iterations in file ITER.OUT.

B. Results

In Figures 1, 2, and 3, test results begin after the lines "*****CALCULATING SIMPLEX*****". Tables 1 and 2 show the latter iterations from iteration history file ITER.OUT produced by executing the cases shown in Figures 1 and 2, respectively. After each execution, this file can be examined, printed, or renamed, but file ITER.OUT is destroyed and recreated near the beginning of each execution.

Figure 1 shows convergence criterion 1 being satisfied with 74 iterations for a selected error limit of 1.D-05. Table 1 shows that all coefficient relative errors are indeed less than 1.D-05 at iteration 74. This table also,shows the construction of this diagnostic file, with results labeled for iteration number 69.

Figure 2 shows the same case as Figure 1 except that convergence criterion 2 was selected. Note that the final value of the function (0.414162E-24) is lower, and the number of iterations (224) is higher. Criterion 2 is often more stringent than criterion 1.

Table 2 shows iterations 211-224 from file ITER.OUT. Notice that beginning with iteration 213, the relative errors develop a pattern as iterations proceed: the first and third ones switch back and forth between pairs of values, and the second remains constant. This continues for 3*(MM+1)=12 iterations, and then criterion 2 is satisfied. This type of error behavior is common when the simplex has locked in to a minimum. Given the logical rules for changing vertex values and the truncation/round-off limits in the computer's arithmetics, the program is unable to find a change which results in a lower function value. Without the logic for criterion 2, iterations would have proceeded until the iteration limit was exceeded.

The case in Figure 3 is the same as that in Figure 2 except the initial estimates were 200, 300, 400 instead of 2, 3, 4, which were much closer to the correct answers 1, 2, 3. Using convergence criterion 2, 268 iterations were required. Approximate wall clock execution time was 30 seconds for Figure 3 and 26 seconds for Figure 2 (224 iterations). For comparison, using single precision, the execution times were approximately 16 and 12 seconds, respectively. However, the estimation of the true function minimum of 0.0 was about 10^{-12} - 10^{13} using single precision, while double precision improved it to approximately 10^{29} - 10^{30}.

IV. CONCLUSIONS

The computer program SIMMIN is capable of finding a minimum (if one exists) for any function which can be described using Fortran. It was tested on an IBM PC with an 8087 math coprocessor using Ryan-McFarland Fortran. This

simplex method determines values of initially unknown coefficients which result in a local function minimum.

The user can vary initial estimates of the coefficients, step sizes, maximum number of iterations, and the convergence criteria, of which two are available. Final values for the coefficients, an estimate of the function minimum, and the number of iterations required are displayed on the screen. A diagnostic iteration history is recorded on a separate file which can be examined after each execution.

The program can easily be converted for use on larger computers with larger word sizes, perhaps allowing the use of single precision. Although divergence is impossible using this simplex method, a price is often paid in more iterations and computing time than with some other traditional minimization methods, such as those mentioned in the introduction.

The user is cautioned against drawing overly optimistic conclusions from the test results. A very simple test function was used. Actual functions encountered in research and development - analytical, empirical, or hybrids - which cause problems when using minimization methods such as those mentioned in the introduction could require care when using simplex techniques as well.

Here are a few advantages simplex methods such as those in SIMMIN offer over other methods:

1. No derivatives of any kind are calculated.

2. Divergence is impossible.

3. Initial estimates for the coefficients can be poor, but this will slow down convergence, depending upon step size.

4. Simplex logic insures economical calculation of the response function.

5. No matrix operations are involved.

6. Constraint conditions which can be stated in Fortran can be added very easily.

Here are a few disadvantages of simplex compared with other methods:

1. More iterations and computing time can be required. SIMMIN is recommended for larger computers, especially when the function is complicated, when other methods have proved unsatisfactory or need to be verified, or when the user has little knowledge of the functional behavior.

2. The following can cause computational problems: very large initial estimates coupled with small steps, insensitivity of the function to one or more coefficients, too large an acceptable error when using an error comparison convergence criterion. These and other problem areas are discussed in much more detail in Reference 2.

REFERENCES

1. Nelder, J. A., Mead, R., "A Simplex Method for Function Minimization", Computer Journal, Vol. 7, pp. 308-313, 1965.

2. Caceci, Marco S., Cacheris, William P., "Fitting Curves to Data", Byte Magazine, pp. 340-362, May 1984.

3. Holloman, Miles E., Otto, William F., "PCFIT2 - General Purpose Curve Fitting Program for IBM PC", MICOM Technical Report RH-85-3, August 1985.

4. Holloman, Miles E., Otto, William F., Smith, Jerry D., "PCFIT Version 2.0 Plotting, Fitting, Interpolating Utility Program for IBM PC", MICOM Technical Report RD-DE-86-3, July 1986.

```
ENTER MM, NUMBER OF PARAMETERS TO BE CALCULATED -[3]

ENTER MAXITR, MAXIMUM NO. OF ITERATIONS, TYPICALLY SEVERAL HUNDRED.  PROGRAM
USUALLY CONVERGES WITHIN 20*(MM**2) ITERATIONS. -[200]

ENTER MM INITIAL ESTIMATES OF PARAMETERS.  BE SURE THE ENTRY ORDER CONFORMS
WITH YOUR DEFINITION OF PARAMETERS COEF(I) IN SUBROUTINE FUNC.  ESTIMATES
USUALLY CAN BE QUITE INACCURATE. -
[2,3,4]

ENTER MULTIPLIER IN STEP CALCS., APPROX. RANGE FROM 0.1 to 0.5.  (STEP =
ESTIMATE * MULTIPLIER) - [.2]

ENTER CONVERGENCE CRITERION OPTION -
1, FOR COMPARISON OF RELATIVE ERRORS WITH INPUT LIMIT.
2, FOR THE CONDITION THAT THE NUMBER OF UNSUCCESSFUL ATTEMPTS TO DECREASE THE
   FUNCTION EXCEEDS 3*(MM+1) -[1]

ENTER LIMIT ON RELATIVE ERRORS, THE SAME FOR ALL COEFFICIENTS.
RECOMMENDED RANGE 1.D-06 to 1.D-04. - [1.D-05]

***** CALCULATING SIMPLEX*****
FINAL VALUES OF COEFFICIENTS -
   1.00000    2.00000    3.00000
FINAL VALUE (MINIMUM) OF FUNCTION = 0.417040E-10
NUMBER OF ITERATIONS = 74
Execution terminated : OK
```

Figure 1. <u>Screen input and results for a typical execution of SIMMIN. Convergence criterion 1 was selected.</u>

TABLE 1. Latter Contents of Iteration History File ITER.OUT Produced During Execution With Input Parameters Shown in Figure 1.

```
69 = Iteration number

1.000
2.000   Current values of coefficients
3.000

0.1210E-09   Estimate of function value, printed only when the iteration led
             to a better (lower) function value.

 1.48453975763265370E-05    7.86123165807031016E-06    7.56428230863572653E-06
70

        Estimates of relative errors for the coefficients, MM = 3

1.0000
 2.000
 3.000
0.7947E-10

 1.48453976763265370E-05    4.89503933556161228E-06    7.56428230863572653E-06
71

1.000
2.000
3.000

 1.44438985795023159E-05    4.89503933556161228E-06    7.56428230863572653E-06
72

1.0000
 2.000
 3.000
0.3374E-10

 1.20481762607656796E-05    3.12815572995717666E-06    5.20121882225269973E-06
73

1.000
2.000
3.000
0.3160E-10

 1.20481762607656796E-05    3.12815572995717666E-06    2.91527801458411470E-06
74

1.000
2.000              All relative errors < 1.D-05.
3.000          Option 1 convergence (ICON = 1) obtained.

 6.93297636726561162E-06    3.12815572995717666E-06    2.91527801458411470E-06
```

```
ENTER MM, NUMBER OF PARAMETERS TO BE CALCULATED - [3]

ENTER MAXITR, MAXIMUM NO. OF ITERATIONS, TYPICALLY SEVERAL HUNDRED.  PROGRAM
USUALLY CONVERGES WITHIN 20*(MM**2) ITERATIONS. - [300]

ENTER MM INITIAL ESTIMATES OF PARAMETERS.  BE SURE THE ENTRY ORDER CONFORMS
WITH YOUR DEFINITION OF PARAMETERS COEF(I) IN SUBROUTINE FUNC.  ESTIMATES
USUALLY CAN BE QUITE INACCURATE. -
[2,3,4]

ENTER MULTIPLIER IN STEP CALCS., APPROX. RANGE FROM 0.1 to 0.5.  (STEP =
ESTIMATE * MULTIPLIER) - [.2]

ENTER CONVERGENCE CRITERION OPTION -
1, FOR COMPARISON OF RELATIVE ERRORS WITH INPUT LIMIT.
2, FOR THE CONDITION THAT THE NUMBER OF UNSUCCESSFUL ATTEMPTS TO DECREASE THE
   FUNCTION EXCEEDS 3*(MM+1) - [2]

***** CALCULATING SIMPLEX*****
FINAL VALUES OF COEFFICIENTS -
   1.00000    2.00000    3.00000
FINAL VALUE (MINIMUM) OF FUNCTION = 0.414162E-29
NUMBER OF ITERATIONS = 224
Execution terminated : OK
```

Figure 2. <u>Screen input and results for a typical execution of SIMMIN. Convergence criterion 2 was selected.</u>

TABLE 2. Latter Contents of File ITER.OUT Produced by Input Parameters in Figure 2. Repetitive Pattern in Coefficient Errors Begins at Iteration 213.

```
211

 1.000
 2.000
 3.000
0.4930E-31

 2.22044604925031111E-15    1.11022302462515536E-15    8.88178419700124936E-16
212

 1.0000
  2.000
  3.000

 2.22044604925031111E-15    6.66133814775093530E-16    5.92118946466750122E-16
213

 1.0000
  2.000
  3.000

 1.99840144432528138E-15    6.66133814775093530E-16    5.92118946466750023E-16
214

 1.000
 2.000
 3.000

 2.22044604925031111E-15    6.66133814776093530E-16    5.92118946466750122E-16
215

 1.0000
  2.000
  3.000

 1.99840144432528138E-15    6.66133814775093530E-16    5.92118946466750023E-16
216

 1.000
 2.000
 3.000

 2.22044604825031111E-15    6.66133814775093530E-16    5.92118946466750122E-16
217

 1.0000
  2.000
  3.000
```

TABLE 2. (Concluded)

```
 1.99840144432528138E-15    6.66133814775093530E-16    5.92118946466750023E-16
218

 1.000
 2.000
 3.000

 2.22044604925031111E-15    6.66133814775093530E-16    5.92118946466750122E-16
219

 1.0000
  2.000
  3.000

 1.99840144432528138E-15    6.66133814775093530E-16    5.92118946466750023E-16
220

 1.000
 2.000
 3.000

 2.22044604925031111E-15    6.66133814775093530E-16    5.92118946466750122E-16
221

 1.0000
  2.000
  3.000

 1.99840144432528138E-15    6.66133814775093530E-16    5.92118946466750023E-16
222

 1.000
 2.000
 3.000

 2.22044604925031111E-15    6.66133814775093530E-16    5.92118946466750122E-16
223

 1.0000
  2.000
  3.000

 1.99840144432528138E-15    6.66133814775093530E-16    5.92118946466750023E-16
224

 1.000
 2.000
 3.000

 2.22044604925031111E-15    6.66133814775093530E-16    5.92118946466750122E-16
```

```
ENTER MM, NUMBER OF PARAMETERS TO BE CALCULATED -[3]

ENTER MAXITR, MAXIMUM NO. OF ITERATIONS, TYPICALLY SEVERAL HUNDRED.
PROGRAM USUALLY CONVERGES WITHIN 20*(MM**2) ITERATIONS. - [300]

ENTER MM INITIAL ESTIMATES OF PARAMETERS.  BE SURE THE ENTRY ORDER CONFORMS
WITH YOUR DEFINITION OF PARAMETERS COEF(I) IN SUBROUTINE FUNC.  ESTIMATES
USUALLY CAN BE QUITE INACCURATE. -
[200,300,400]

ENTER MULTIPLIER IN STEP CALCS., APPROX. RANGE FROM 0.1 to 0.5.  (STEP =
ESTIMATE * MULTIPLIER) - [.2]

ENTER CONVERGENCE CRITERION OPTION -
1, FOR COMPARISON OF RELATIVE ERRORS WITH INPUT LIMIT.
2, FOR THE CONDITION THAT THE NUMBER OF UNSUCCESSFUL ATTEMPTS TO DECREASE THE
   FUNCTION EXCEEDS 3*(MM+1) -[2]

***** CALCULATING SIMPLEX*****
FINAL VALUES OF COEFFICIENTS -
   1.00000    2.00000    3.00000
FINAL VALUE (MINIMUM) OF FUNCTION = 0.246519E-30
NUMBER OF ITERATIONS = 268
Execution terminated : OK
```

Figure 3. Screen input and results for a typical execution of SIMMIN. Initial coefficient estimates were 100, 200, 300. Convergence criterion 2 was selected.

APPENDIX

FORTRAN SOURCE CODE FOR SIMMIN

```
                  PROGRAM SIMMIN
C
C     PROGRAM SIMMIN CAN  FIND A MINIMUM (IF ONE EXISTS) FOR ANY
C     FUNCTION DEFINED BY THE USER IN SUBROUTINE FUNC.  THE FUNCTION
C     CAN BE LINEAR OR NONLINEAR, PARTIALLY EMPIRICALLY DEFINED, AND
C     NEED NOT BE CONTINUOUS. SIMMIN DOES THIS BY USING SIMPLEX
C     TECHNIQUES TO CALCULATE THE VALUES OF INITIALLY UNKNOWN
C     COEFFICIENTS THAT YIELD THE MINIMUM.  PROGRAM IS BASED ON A
C     PASCAL ALGORITHM IN "FITTING CURVES TO DATA", BYTE, MAY 1984,
C     WHICH WAS CONVERTED TO FORTRAN77 AND MODIFIED FOR USE IN
C     GENERAL PC CURVE FITTING PROGRAMS BY MILES HOLLOMAN AND BILL OTTO,
C     DIRECTED ENERGY DIRECTORATE, MICOM.  FURTHER MODIFICATIONS
C     RESULTED IN THIS INITIAL VERSION OF SIMMIN BY JERRY D. SMITH,
C     OF THE SAME DIRECTORATE. USER SHOULD BE FAMILIAR ENOUGH WITH
C     FORTRAN TO MODIFY SUBROUTINE FUNC AS REQUIRED, CONVERGENCE
C     CRITERIONS IF DESIRED (SEE COMMENTS WITHIN SUBROUTINE SIMPLX),
C     AND PERHAPS MINOR FORMAT CHANGES IF MANY UNKNOWN COEFFICIENTS
C     COEF(I) OCCUR IN THE FUNCTION. CURRENT MAX. NO. = 10.
C     NOTE --- EXECUTION TIMES INCREASE GREATLY WITH NUMBER OF
C     COEFFICIENTS, ABOVE THREE.
C
C     AFTER EXECUTION FILE ITER.OUT WILL CONTAIN A HISTORY OF THE
C     ITERATION RESULTS : COEFFICIENT AND FUNCTION VALUES, AND RELATIVE
C     ERROR VALUES FOR THE COEFFICIENTS.  USER CAN EXAMINE AFTER EXECU-
C     TION IF DESIRED.  IT IS DESTROYED AND RECREATED DURING THE NEXT
C     EXECUTION.
C
C     ADVANTAGES OVER OTHER NONLINEAR METHODS:
C     1. NO DERIVATIVES ARE CALCULATED.
C     2. DIVERGENCE IS IMPOSSIBLE.
C     3. INITIAL ESTIMATES FOR COEFFICIENTS CAN BE QUITE POOR,
C        ALTHOUGH THIS WILL SLOW DOWN CONVERGENCE.
C     4. SIMPLEX LOGIC INSURES ECONOMICAL CALCULATIONS OF RESPONSE
C        FUNCTION.
C     5. NO MATRIX OPERATION IS INVOLVED.
C     6. CONSTRAINTS CAN BE ADDED VERY EASILY. (SEE SUBROUTINE FUNC.)
C
C     DISADVANTAGES OVER OTHER NONLINEAR METHODS:
C     1. MORE ITERATIONS AND COMPUTING TIME ARE OFTEN REQUIRED. SIMMIN
C        RECOMMENDED FOR FAST PC'S, AND NUMBER-CRUNCHERS, ESPECIALLY
C        WHEN THE FUNCTION TO BE MINIMIZED IS COMPLICATED AND USER HAS
C        VERY LITTLE IF ANY KNOWLEDGE OF THE FUNCTIONAL BEHAVIOR.
C     2. FOLLOWING CAN CAUSE COMPUTATIONAL PROBLEMS, AND RESULT IN
C        INACCURATE ANSWERS: VERY LARGE INITIAL GUESSES, VERY SMALL
C        INITIAL INCREMENTS, INSENSITIVITY OF FUNCTION TO ONE OR MORE
C        COEFFICIENTS, TOO LARGE AN ACCEPTABLE ERROR (WHEN USING
C        CONVERGENCE OPTION ICON=2).
C
C
C                          JERRY D. SMITH
C                          11 MAY 87
C
C
      COMMON FINAL(11),SIMP(11,11),STEP(11),MM,LUO,MAXITR,NITER,ICON,RX
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
C
```

```
      LUO=7
      OPEN(UNIT=LUO,FILE='ITER.OUT',STATUS='UNKNOWN')
      CLOSE(UNIT=LUO,STATUS='DELETE')
      OPEN(UNIT=LUO,FILE='ITER.OUT',STATUS='NEW')
C
      WRITE(*,1600)
 1600 FORMAT(1H1,'ENTER MM, NUMBER OF PARAMETERS TO BE CALCULATED - ')
      READ(*,*) MM
      WRITE(*,1601)
 1601 FORMAT(1X,'ENTER MAXITR, MAXIMUM NO. OF ITERATIONS,TYPICALLY SEVER
     XAL HUNDRED.',/' PROGRAM USUALLY CONVERGES WITHIN 20*(MM**2) ITERAT
     XIONS. - ')
      READ(*,*) MAXITR
      MM1=MM+1
      WRITE(*,1602)
 1602 FORMAT(1X,'ENTER MM INITIAL ESTIMATES OF PARAMETERS. BE SURE'
     X,/,' THE ENTRY ORDER CONFORMS WITH YOUR DEFINITION OF PARAMETERS'
     X,/,' COEF(I) IN SUBROUTINE FUNC. ESTIMATES USUALLY CAN BE QUITE'
     X,/,' INACCURATE - ')
      READ(*,*) (SIMP(I,1),I=1,MM)
      WRITE(*,1603)
 1603 FORMAT(1X,'ENTER MULTIPLIER IN STEP CALCS., APPROX.RANGE FROM 0.1
     X',/,' TO 0.5.  (STEP=ESTIMATE * MULTIPLIER) - ')
      READ(*,*) SMULT
      DO 10 J=1,MM
   10 STEP(J)=SIMP(J,1)*SMULT
C
      PRINT 1610
 1610 FORMAT(1X, 'ENTER CONVERGENCE CRITERION OPTION -',/1X, '1, FOR COM
     XPARISON OF RELATIVE ERRORS WITH INPUT LIMIT.',/1X, '2, FOR THE CON
     XDITION THAT THE NUMBER OF UNSUCCESSFUL ATTEMPTS TO',/1X, '   DECRE
     XASE THE FUNCTION EXCEEDS 3*(MM+1) - ')
      READ(*,*) ICON
      IF(ICON.EQ.1) THEN
      PRINT 1612
 1612 FORMAT(1X, 'ENTER LIMIT ON RELATIVE ERRORS, THE SAME FOR ALL COEFF
     XICIENTS.  ',/1X, 'RECOMMENDED RANGE 1.D-06 TO 1.D-04 -')
      READ(*,*) RX
      ENDIF
C
      CALL SIMPLEX
C
      WRITE(*,1604)  (FINAL(I),I=1,MM)
 1604 FORMAT(1X,'FINAL VALUES OF COEFFICIENTS - ',/3X, 6G12.6)
      PRINT 1606, FINAL(MM1)
 1606 FORMAT(   ' FINAL VALUE (MINIMUM) OF FUNCTION = ', G12.6)
      PRINT 1608, NITER
 1608 FORMAT(1X,'NUMBER OF ITERATIONS=',I4)
C
      STOP  'OK'
      END
```

```
      SUBROUTINE SIMPLEX
C
      LOGICAL DONE,LEQ
      COMMON FINAL(11),SIMP(11,11),STEP(11),MM,LUO,MAXITR,NITER,ICON,RX
      DIMENSION CENTER(11),ERROR(11),RMAXER(11),PS(11),Q(11),IH(11)
      DIMENSION RNEXT(11),L(11)
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
C
      DLOW=1.0D+38
      DLOWL=1.0D+38
      LCNT=0
      PRINT 1030
 1030 FORMAT(' ***** CALCULATING SIMPLEX ***** ')
C
      MM1=MM+1
      DO 1040 I=1,11
 1040 RMAXER(I)=RX
C
C     FOLLOWING ARE PARAMETERS TO VARY THE SIMPLEX MODIFICATIONS AND
C     SHOULD GENERALLY BE ACCEPTED AS OPTIMAL VALUES.
C
      ALFA = 1.0
      BETA = 0.5
      GAMMA = 2.0
C
C
      TWO = 2.0
      ROOT2 = DSQRT(TWO)
      CALL FUNC(SIMP(1,1),MM1)
      DO 1060 I=1,MM
      RNN = MM1
      PS(I)=STEP(I)*((SQRT(RNN)+MM-1))/(MM*ROOT2)
      Q(I)=STEP(I)*((SQRT(RNN))-1)/(MM*ROOT2)
 1060 CONTINUE
      DO 1080 I=2,MM1
      DO 1070 J=1,MM
 1070 SIMP(J,I)=SIMP(J,1)+Q(J)
      SIMP(I-1,I)=SIMP(I-1,1)+PS(I-1)
      CALL FUNC(SIMP(1,I),MM1)
 1080 CONTINUE
      DO 1090 I=1,MM1
      L(I)=1
 1090 IH(I)=1
      CALL ORDER(MM1,SIMP,L,IH)
      NITER=0
 1100 DONE=.TRUE.
      NITER=NITER+1
C
C     COMPUTE CENTROID OF THE SIMPLEX.
C
      DO 1110 I=1,MM1
 1110 CENTER(I)=0.0
      DO 1130 I=1,MM1
      IF(I.EQ.IH(MM1)) GO TO 1130
      DO 1120 J=1,MM
      CENTER(J)=CENTER(J)+SIMP(J,I)
```

```
 1120  CONTINUE
 1130  CONTINUE
C
C      TRY SPECULAR REFLECTION.
C
       DO 1140 I=1,MM1
       CENTER(I)=CENTER(I)/MM
       RNEXT(I)=(1.0+ALFA)*CENTER(I)-ALFA*SIMP(I,IH(MM1))
 1140  CONTINUE
       CALL FUNC(RNEXT,MM1)
C
       IF(RNEXT(MM1).LE.SIMP(MM1,L(MM1))) GO TO 1150
       GO TO 1170
 1150  CALL NEWVR(MM1,SIMP,RNEXT,IH)
C
C      TRY AN EXPANSION.
C
       DO 1160 I=1,MM
       RNEXT(I)=GAMMA*SIMP(I,IH(MM1))+(1.0-GAMMA)*CENTER(I)
 1160  CONTINUE
       CALL FUNC(RNEXT,MM1)
C
       IF(RNEXT(MM1).LE.SIMP(MM1,L(MM1)))
      X CALL NEWVR(MM1,SIMP,RNEXT,IH)
       GO TO 1250
C
 1170  IF(RNEXT(MM1).LE.SIMP(MM1,IH(MM1)))  GO TO 1180
       GO TO 1190
 1180  CALL NEWVR(MM1,SIMP,RNEXT,IH)
       GO TO 1250
C
C      TRY A CONTRACTION.
C
 1190  DO 1200 I=1,MM
 1200  RNEXT(I)=BETA*SIMP(I,IH(MM1))+(1.0-BETA)*CENTER(I)
       CALL FUNC(RNEXT,MM1)
C
       IF(RNEXT(MM1).LE.SIMP(MM1,IH(MM1)))  GO TO 1210
       GO TO 1220
 1210  CALL NEWVR(MM1,SIMP,RNEXT,IH)
       GO TO 1250
C
C      DO A SHRINKAGE.
C
 1220  DO 1240 I=1,MM1
       DO 1230 J=1,MM
 1230  SIMP(J,I)=(SIMP(J,I)+SIMP(J,L(MM1)))*BETA
 1240  CALL FUNC(RNEXT,MM1)
 1250  CONTINUE
 1270  FORMAT(I4/)
 1280  FORMAT(G12.4)
       IF(LEQ)  LCNT=LCNT+1
       IF(.NOT.LEQ) LCNT=0
       DLOWL=DLOW
C
       WRITE(LUO,1270)  NITER
```

```
      DO 1308 I=1,MM1
      IF(I.NE.MM1)  WRITE(LUO,1280) RNEXT(I)
      IF(I.EQ.MM1.AND.RNEXT(MM1).LT.DLOW)  WRITE(LUO,1280) RNEXT(I)
 1308 FINAL(I)=RNEXT(I)
      WRITE(LUO,1420)
      IF(RNEXT(MM1).LT.DLOW) DLOW=RNEXT(MM1)
C
 1305 LEQ=(DLOWL.EQ.DLOW)
      CALL ORDER(MM1,SIMP,L,IH)
C
      DO 1320 J=1,MM
      ERROR(J)=(SIMP(J,IH(J))-SIMP(J,L(J)))/SIMP(J,IH(J))
 1320 CONTINUE
      WRITE(LUO,*) (ERROR(J),J=1,MM)
C
C     FOLLOWING IS PRESENT AUTHORS' CRITERION FOR CONVERGENCE,
C     BASED ON EXPERIENCE WITH NONLINEAR FUNCTIONS IN FITTING
C     APPLICATIONS. ESSENTIALLY, THIS CRITERION SAYS THAT IF NUMBER OF
C     CONSECUTIVE UNSUCCESSFUL ATTEMPTS  FIND A NEW DIRECTION AND
C     SHAPE TO DECREASE THE VALUE OF THE FUNCTION REACHES MM1*3,
C     THEN CALL IT A GAME AND RETURN TO MAIN PROGRAM. THE SIMPLEX HAS
C     EITHER FOUND A MINIMUM AND IS SPINNING ITS WHEELS, OR IT IS EATING
C     UP TIME DUE TO POORLY CHOSEN INPUTS. TRY AGAIN WITH DIFFERENT
C     INITIAL ESTIMATES AND STEP SIZES, IF USER SUSPECTS SOMETHING
C     UNUSUAL. IN FACT, IT IS ALWAYS GOOD PROCEDURE TO VERIFY
C     RESULTS BY RERUNNING WITH DIFFERENT INPUTS, AND TO INSPECT
C     ITERATION HISTORY FILE ITER.OUT.
C
      IF(ICON.EQ.2 .AND. LCNT.GE.MM1*3)  GO TO 1350
C
C     THE FOLLOWING IS A DIFFERENT TYPE OF CONVERGENCE CRITERION.
C
      IF(ICON.EQ.1) THEN
      K=0
      DO 1322 J=1,MM
      IF(ERROR(J).LT.RMAXER(J)) K=K+1
 1322 CONTINUE
      IF(K.GE.MM) GO TO 1350
      ENDIF
      IF(NITER.GT.MAXITR) GO TO 1350
C
C     TRY AGAIN.
C
      GO TO 1100
C
C     CONVERGENCE OBTAINED, OR ITERATION LIMIT EXCEEDED, ETC.
C
 1350 CONTINUE
      WRITE(LUO,1420)
 1420 FORMAT(' ')
      RETURN
      END
```

```
      SUBROUTINE FUNC(COEF,MM1)
C
C     A SIMPLE TEST FUNCTION, WITH THREE UNKNOWN COEFFICIENTS
C     COEF(J), J=1,3, IS TO BE MINIMIZED.  NOTE THE CURRENT FUNCTION
C     VALUE (THE MM+1 th VERTEX OF THE SIMPLEX) IS STORED IN COEF(MM+1)
C
      DIMENSION COEF(1)
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
C
      TERM1=COEF(1)-1.0
      TERM2=COEF(2)-2.0
      TERM3=COEF(3)-3.0
      COEF(MM1)=TERM1**2 + TERM2**2 + TERM3**2
      RETURN
C
C     CONSTRAINT CONDITION EXCEEDED - SEND BACK A MESSAGE TO
C     SUBROUTINE SIMPLX, BY TAGGING ON A LARGE VALUE TO THE FUNCTION.
C     NOTE ---- USER MUST PROTECT AGAINST SUCH OCCURRENCES AS TRYING
C     TO TAKE LOGARITHMS OF 0 OR NEGATIVE NUMBERS, ETC.  TREAT THEM
C     AS CONSTRAINTS, AND DIRECT PROGRAM TO STATEMENT 999, WHEN
C     NECESSARY.
C
  999 CONTINUE
      COEF(MM1)=COEF(MM1)+1.D30
      RETURN
      END
```

```
C
      SUBROUTINE NEWVR(MM1,SIMP,RNEXT,IH)
C
C     THIS SUBROUTINE STORES THE NEW CHOICES FOR VERTICES, WHICH HAVE
C     RESULTED IN A DECREASE IN THE VALUE OF THE RESPONSE FUNTION.
C
      DIMENSION SIMP(11,1),RNEXT(1),IH(1)
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
C
      DO 3010 I=1,MM1
      SIMP(I,IH(MM1))=RNEXT(I)
 3010 CONTINUE
      RETURN
      END
```

```
C
      SUBROUTINE ORDER(MM1,SIMP,L,IH)
C
C     THIS SUBROUTINE DETERMINES CURRENT LOWEST AND HIGHEST VERTICES,
C     AND STORES ORDER INFORMATION USING SECOND INDEX OF SIMP(J,I).
C     THIS WILL BE USED BY SUBROUTINE SIMPLX IN DETERMINING THE NEXT
C     TRIAL SHAPE AND LOCATION OF THE SIMPLEX.
C
      DIMENSION SIMP(11,1),L(1),IH(1)
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
C
      DO 4020 J=1,MM1
      DO 4010 I=1,MM1
      IF(SIMP(J,I).LT.SIMP(J,L(J))) L(J)=I
      IF(SIMP(J,I).GT.SIMP(J,IH(J))) IH(J)=I
 4010 CONTINUE
 4020 CONTINUE
      RETURN
      END
```

DISTRIBUTION

	No. of Copies
IIT Research Institute ATTN: GACIAC 10 W. 35th Street Chicago, IL 60616	1
US Army Materiel System Analysis Activity ATTN: AMXSY-MP Aberdeen Proving Ground, MD 21005	1
AMSMI-RD, Dr. McCorkle Dr. Rhoades	1
AMSMI-RD-CS-T, Record Copy	1
AMSMI-RD-CS-R, Reference	15
AMSMI-RD-DE, Jerry D. Smith	50
AMSMI-GC-IP, Mr. Bush	1

END
DATE
FILMED
8-88
DTIC